

SERIAL NO. 09/542,189

Amendment dated June 11, 2003

Reply to Office Action of Feb. 11, 2003

PATENT
Docket RAL920000008US1**Amendments to the Claims:**

- AM
- 3 1. (currently amended) The use of multiple threads in association with a network
4 processor and accessible data available in a tree search structure, including the
5 steps of:
 - 6 a) providing multiple instruction execution threads as independent processes
7 in a sequential time frame;
 - 8 b) ~~queueing~~ queuing the multiple execution threads to have overlapping access
9 to the accessible data available in said tree search structure;
 - 10 c) executing a first thread in the queue; and
 - 11 d) transferring control of the execution to the next thread in the queue upon the
12 occurrence of an event that causes execution of the first thread to stall.
 - 1 2. (original) The use of the multiple threads according to claim 1 wherein the control
2 of the execution is temporarily transferred to the next thread when execution stalls
3 due to a short latency event, and the control is returned to the original thread when
4 the event is completed.
 - 1 3. (original) The use of multiple threads according to claim 2 wherein a processor
2 instruction is encoded to select a short latency event.
 - 1 4. (original) The use of the multiple threads according to claim 1 wherein full control
2 of the execution is transferred to the next thread when execution of the first thread
3 stalls due to a long latency event.

SERIAL NO. 09/542,189

Amendment dated June 11, 2003

Reply to Office Action of Feb. 11, 2003

PATENT
Docket RAL920000008US1

- A4
- 1 5. (original) The use of multiple threads according to claim 4 wherein a processor
2 instruction is encoded to select a long latency event.
 - 1 6. (currently amended) The use of multiple threads according to claim 1 including
2 ~~queueing~~ queuing the threads to provide rapid distribution of access to shared
3 memory.
 - 1 7. (original) The use of the multiple threads according to claim 1 wherein the threads
2 have overlapping access to shared remote storage via a pipelined coprocessor by
3 operating within different phases of a pipeline of the coprocessor.
 - 1 8. (original) The use of the multiple threads according to claim 1 further including the
2 step of providing a separate instruction pre-fetch buffer for each execution thread,
3 and collecting instructions in a prefetch buffer for its execution thread when the
4 thread is idle and when the instruction bandwidth is not being fully utilized.
 - 1 9. (original) The use of the multiple threads according to claim 1 wherein the threads
2 are used with zero overhead to switch execution from one thread to the next.
 - 1 10. (original) The use of the multiple threads according to claim 9 wherein each thread
2 is given access to general purpose registers and local data storage to enable
3 switching with zero overhead.
 - 1 11. (currently amended) A network processor that uses multiple threads to access
2 data, including:
3 a) a CPU configured with multiple instruction execution threads as independent

SERIAL NO. 09/542,189

Amendment dated June 11, 2003

Reply to Office Action of Feb. 11, 2003

PATENT
Docket RAL920000008US1

- A⁴
- 4 processes in a sequential time frame;
- 5 b) a thread execution control for
- 6 1) queueing queuing the multiple execution threads to have overlapping
- 7 access to the accessible data;
- 8 2) executing a first thread in the queue; and
- 9 3) transferring control of the execution to the next thread in the queue
- 10 upon the occurrence of an event that causes execution of the first
- 11 thread to stall.
- 1 12. (original) A processing system utilizing multiple threads according to claim 11
2 wherein the thread execution control includes control logic for temporarily
3 transferring the control to the next thread when execution stalls due to a short latency
4 event, and for returning control to the original thread when the latency event is
5 completed.
- 1 13. (currently amended) The processing system according to claim 11 12 wherein a
2 processor instruction is encoded to select a short latency event.
- 1 14. (original) The processing system according to claim 11 wherein the control
2 transfer means includes the means for transferring full control of the execution to the
3 next thread when execution of the first thread stalls due to a long latency event.
- 1 15. (original) The processing system according to claim 14 wherein a processor
2 instruction is encoded to select a long latency event.

SERIAL NO. 09/542,189

Amendment dated June 11, 2003

Reply to Office Action of Feb. 11, 2003

PATENT
Docket RAL920000008US1

- A4
- 1 16. (original) The processing system according to claim 11 including means to queue
2 the threads to provide rapid distribution of access to shared memory.
 - 1 17. (original) The processing system according to claim 16 wherein the threads have
2 overlapping access to shared remote storage via a pipelined coprocessor by
3 operating within different phases of a pipeline of the coprocessor.
 - 1 18. (original) The processing system according to claim 11 further including a
2 separate instruction pre-fetch buffer for each execution thread, and means for
3 collecting instructions in a prefetch buffer for an idle execution thread when the
4 instruction bandwidth is not being fully utilized.
 - 1 19. (original) The system according to claim 11 wherein the processor uses zero
2 overhead to switch execution from one thread to the next.
 - 1 20. (original) The system according to claim 19 wherein each thread is given access
2 to an array of general purpose registers and local data storage to enable switching
3 with zero overhead.
 - 1 21. (original) The system according to claim 20 wherein the general purpose
2 registers and the local data storage are made available to the processor by
3 providing one address bit under the control of the thread execution control logic and
4 by providing the remaining address bits under the control of the processor.
 - 1 22. (original) The system according to claim 20 wherein the processor is capable of
2 simultaneously addressing multiple register arrays, and the thread execution control

SERIAL NO. 09/542,189

Amendment dated June 11, 2003

Reply to Office Action of Feb. 11, 2003

PATENT
Docket RAL920000008US1

A 4 3 logic includes a selector to select which array will be delivered to the processor for
4 a given thread.

1 23. (original) The system according to claim 20 wherein the local data storage is fully
2 addressable by the processor, an index register is contained within the register
3 array, and the thread execution control has no address control over the local data
4 storage or the register arrays.

1 24. (currently amended) A network processor configuration comprising:
2 a CPU with multiple threads;
3 an instruction memory, and a separate prefetch queue for each thread
4 between the instruction memory and the CPU;
5 an array of general purposes registers, said array communicating with the
6 CPU;
7 a local data storage including separate storage space for each thread;
8 a thread execution control for the general register array and the local data
9 storage;
10 a first coprocessor connecting the CPU to a local data storage, storage;
11 said thread execution control including priority FIFO buffer to store thread numbers;
12 a shared remote storage; and
13 a pipelined coprocessor connecting the shared remote and the CPU.

1 25. (currently amended) The processor configuration according to claim 24 wherein
2 the thread execution control further includes a priority FIFO, a plurality of thread
3 control state machines, one for each execution thread, and an arbiter responsive to
4 signals from the FIFO buffer and the state machine to determine thread execution

SERIAL NO. 09/542,189

Amendment dated June 11, 2003

Reply to Office Action of Feb. 11, 2003

PATENT
Docket RAL920000008US15 priority.

- 1 26. (original) The use of prefetch buffers in connection with a plurality of independent
2 instruction threads used to process data in a Network Processor comprising the
3 steps of:
4 a) associating each thread with a prefetch buffer;
5 b) determining whether a buffer associated with an execution thread is
6 full;
7 c) determining whether the thread associated with the buffer is active;
8 and
9 d) during periods that the buffer is not being used by an active execution
10 thread, enabling the buffer to prefetch instructions for the execution
11 thread.
- 1 27. (currently amended) A thread execution control useful for the efficient execution of
2 independent threads in a network processor comprising:
3 a) a priority FIFO buffer for storing thread numbers;
4 b) a plurality of thread control state machines, one for each thread; and
5 c) an arbiter for determining the thread execution priority among multiple
6 threads based upon signals outputted from the FIFO buffer and the
7 state machines.
- 1 28. (currently amended) The thread execution control according to claim 27 wherein
2 the FIFO includes:
3 a) means for loading a thread number into the FIFO when a packet is
4 dispatched to the processor;

A4
SERIAL NO. 09/542,189

Amendment dated June 11, 2003

Reply to Office Action of Feb. 11, 2003

PATENT
Docket RAL920000008US1

- 5 b) means for unloading a thread number from the FIFO when a packet
6 has been enqueued for transmission;
7 (c) thread number transfer from highest priority to lowest priority in the
8 FIFO when a long latency event occurs, and
9 (d) the thread outlets of the FIFO used to determine priority depending
10 on the length of time a thread has been in FIFO.

- 1 29. (original) The thread execution control according to claim 27 wherein the arbiter
2 controls the priority of execution of multiple independent threads based on the
3 Boolean expression:

4 — — — —
5 $G_n = R_n \cdot \{(P_A = n) + R_{PA} \cdot (P_B = n) + R_{PA} \cdot R_{PB} \cdot (P_C = n) \dots\}$

6 where: G is a grant

7 R_n is a request from a given thread;

8 P_A, P_B and P_C represent threads ranked by alphabetical subscript
9 according to priority;

10 n is a subscript identifying a thread by the bit or binary number

11 comprising

- 12 a) determining whether a request R is active or inactive;
13 b) determining the priority of the threads;
14 c) matching the request R with the corresponding thread P; and
15 d) granting a request for execution if the request is active and if
16 the corresponding thread P has the highest priority.

SERIAL NO. 09/542,189

Amendment dated June 11, 2003

Reply to Office Action of Feb. 11, 2003

PATENT
Docket RAL920000008US1

- AM*
- 1 30. (currently amended) The thread execution control according to claim 27 wherein
2 the thread control state
3 machine comprises control logic to :
4 (a) dispatch a packet to a thread;
5 (b) move the thread from an initialize state to a ready state;
6 (c) request execution cycles for the thread packet;
7 (d) move the thread to the execute state upon grant by the arbiter of an
8 execution cycle;
9 (e) continue to request execution cycles while the thread packet is
10 queued in the execute state; and
11 (f) return the thread packet to the initialize state if there is no latency
12 event, or send the packet to the wait state upon occurrence of a
13 latency event.

31. (New) The thread executing control according to claim 28 wherein the FIFO
further includes means to detect occurrence of latency events.